High-Level Area and Power Estimation for VLSI Circuits†

Mahadevamurty Nemani and Farid N. Najm
ECE Dept. and Coordinated Science Lab.
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Abstract

This paper addresses the problem of computing the area complexity of a multi-output combinational logic circuit, given only its functional description, i.e., Boolean equations, where area complexity is measured in terms of the number of gates required for an optimal multi-level implementation of the combinational logic. The proposed area model is based on transforming the given multi-output Boolean function description into an equivalent single-output function. The model is empirical, and results demonstrating its feasibility and utility are presented. Also, a methodology for converting the gate count estimates, obtained from the area model, into capacitance estimates is presented. High-level power estimates based on the total capacitance estimates and average activity estimates are also presented.

1. Introduction

Rapid increase in the design complexity and reduction in design time have resulted in a need for CAD tools that can help make important design decisions *early* in the design process. To do so, these tools must operate with a design description at a high level of abstraction. One design criterion that has received increased attention lately is power dissipation. This is due to the increasing demand for low power mobile and portable electronics. As a result, there is a need for *high level power estimation* and optimization. Specifically, it would be highly beneficial to have a power estimation capability, given only a *functional* view of the design, such as when a circuit is described only with Boolean equations. In this case, no structural information is known - the lower-level (gate-level or lower) description of this function is not available. Of course, a given Boolean function can be implemented in many ways, with varying power dissipation levels. We are interested in predicting the *nominal* power dissipation that a minimal area implementation of the function would have.

For a combinational circuit, since the only available information is its Boolean function, we consider that its power dissipation will be modeled as follows:

$$P_{avg} \propto \mathcal{D}_{avg}\mathcal{A}\mathcal{C}_{avg} \qquad (1)$$

where $\mathcal{D}_{avg}$ is an estimate of the average node switching activity that a gate-level implementation of this circuit would have, $\mathcal{A}$ is an estimate of the gate count (assuming some target gate library), and $\mathcal{C}_{avg}$ is an estimate of the average node capacitance (including drain capacitance and interconnect loading capacitance). The estimation of $\mathcal{D}_{avg}$ was covered in [1-3]. The problem of estimating $\mathcal{A}$ from a high-level description of the circuit corresponds to the problem of high-level area estimation. This problem is of independent interest, as the information it provides can be very useful, for instance, during floorplanning. The estimation of gate count (or simply, area) $\mathcal{A}$ of single-output Boolean functions was explored in [4, 5], where the problem was addressed using a notion of complexity of the on-set and the off-set of a Boolean function. In this paper the authors propose an area model to predict the area complexity of multi-output Boolean functions. This area model is based on a transformation, which transforms the given multi-output Boolean function into an equivalent single-output function. The transformation is such that it helps us infer the area complexity of the multi-output Boolean function from the area complexity of the single-output function, thus enabling the utilization of the complexity based area model of [5], developed for single-output functions.

However, the proposed area model, like its single-output counterparts [4, 5], is inherently limited to circuits which do not have large exclusive-or arrays in them. Circuits with large exclusive-or arrays are also the source of problems in other CAD areas, such as BDD construction for verification. One way around the problem of exclusive-or arrays is to require that the Boolean function specification explicitly list exclusive-or gates. In that case, these can be identified up-front and excluded from the analysis, so that the proposed method is applied only to the remaining circuitry. In any case, in the remainder of this paper we will not consider circuits composed of large exclusive-or arrays.

Before leaving this section, we should mention some previous work on layout area estimation from an RTL view. Wu et. al. [6] proposed a layout area model for datapath and control for two commonly used layout architectures based on the transistor count. For datapath units, the average transistor count was obtained by averaging the number of transistors over different implementations and, for control logic, they calculate the number of transistors from the sum of products (SOP) expression for the next state and control signals.

† This work was supported in part by Intel Corp., by Rockwell, and by an NSF CAREER Award (MIP-9623237).

A similar model was proposed by Kurdahi et.al. [7]. Both these models consider the effect of interconnect on the overall area, while [7] considers the effect of cell placement on the overall area. Since the controller area, in [6, 7], is estimated based on the number of AND and OR gates required to implement the SOP expression, the optimal number of gates required to implement the function can be much smaller than the above sum. This is because it is frequently possible to apply logic optimization algorithms to give a much better implementation.

2. The Multi-Output Area Model

We aim to estimate the *minimum* number of gates ($\mathcal{A}$) required for a multi-level implementation the function, given only its high level description (Boolean equations) and a target technology library. The area model proposed for single-output Boolean functions [4, 5] is based on the notion of complexity of the on and off-sets of a Boolean function. One such complexity measure which will be used in this paper is the *linear measure*, defined in [5].

Our approach to solving the multi-output area estimation problem is inspired by the multi-valued logic approach to address the problem of two-level minimization of multi-output Boolean functions [8]. The approach is based on transforming a binary-valued, multi-output Boolean function into an equivalent multi-valued-input, single-output (binary-valued) Boolean function. The transformation is accomplished by adding an m-valued input to the Boolean function. Each value of the multi-valued input corresponds to one of the original m outputs. In our approach we perform a similar transformation, except that we use $\lceil \log_2(m) \rceil$ binary-valued inputs to implement an m-valued input. An equivalent way of representing the transformation is to think of the additional $\lceil \log_2(m) \rceil$ binary-valued inputs as control signals of a multiplexor, and that the value of the control word corresponds to the output being selected. This corresponds to multiplexing the m outputs of a m-output Boolean function, as shown if Fig. 1. The original n input m output function f is thus transformed to a $(n + \lceil \log_2 m \rceil)$ input, single-output Boolean function $\hat{f}$. Since $\hat{f}$ is a single-output function, its area can be computed by the application of the area model of [5]. It must be noted that since we are applying the area estimation technique to $\hat{f}$, which is made up of all the outputs, we are in effect dealing with all the outputs at the same time and thus automatically accounting for the effect of sharing.

A natural question to ask is, what is the relation between the (optimal) area of f and that of $\hat{f}$. To answer this question, consider the following two scenarios. In the first scenario, let all the outputs of the multi-output Boolean function be the same. In this case the area of the multi-output Boolean function is equal to the area of any of its outputs. Also note that the prime implicants of the on and off-sets of $\hat{f}$ are independent of the control inputs. Hence the complexity measure of $\hat{f}$ is equal to the complexity measure of any of the outputs of f. Also, as all the outputs of the function are the same, there is no need for the multiplexor. Thus the area contribution of the multiplexor to the overall area of a minimized $\hat{f}$ is zero. Now consider the second scenario. Here, assume that all the outputs of the multi-output Boolean function have disjoint support sets. It then follows that the optimal area of $\hat{f}$ is equal to the sum of the optimal area complexity of f and the area complexity of the multiplexor. Thus one has to subtract the area of the multiplexor from the area complexity of $\hat{f}$ in order to get the area complexity of f. Moreover every prime implicant in the on and off-sets of $\hat{f}$ contains all the control inputs.

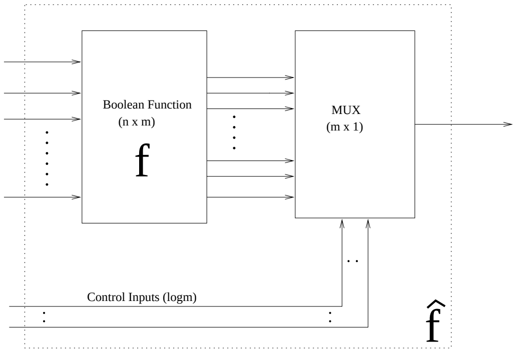

Figure 1. Transformation of a m output Boolean function into a single output Boolean function.

In the first scenario, when the contribution of the multiplexor to the area of $\hat{f}$ was zero, we saw that the control inputs were absent from all the prime implicants, while in the second scenario when the contribution of the multiplexor to the area of $\hat{f}$ is maximum, we saw that all the control inputs are present in every prime implicant of $\hat{f}$. Thus there seems to be a correlation between the influence of the multiplexor on the area of $\hat{f}$ and the number of control inputs in the prime implicants of $\hat{f}$.

The difference $\mathcal{A}(\hat{f}) - \mathcal{A}(f)$ represents the area contribution of the multiplexor to an optimal area implementation of $\hat{f}$. Note that after optimization it might so happen that certain control inputs become redundant for certain outputs. This manifests itself as some control inputs being absent in some prime implicants of on and off-sets of $\hat{f}$. Thus, we may think of $\mathcal{A}(\hat{f}) - \mathcal{A}(f)$ as representing the area of a *reduced* multiplexor resulting from the optimization. This reduced multiplexor area is related to the number of remaining control signals, which leads us to a method for estimating this area, as follows.

From the above considerations, we propose that an appropriate area model for a multi-output function f, in terms of the area of $\hat{f}$ and the area of a m to 1 multiplexor is given by

$$\mathcal{A}(f) = \mathcal{A}(\hat{f}) - \alpha \mathcal{A}_{mux} \qquad (2)$$

where $\mathcal{A}_{mux}$ is the area complexity of an m to 1 multiplexor, and $0 \leq \alpha \leq 1$ is a coefficient that represents the contribution of the multiplexor to the area complexity of $\hat{f}$. In the following, we present an approach for estimating α.

Note that the complexity measure [5] of a m to 1 multiplexor is given by $\lceil \log_2 m \rceil + 1$, i.e., the complexity of a m to 1 multiplexor is proportional to the number of control inputs. This is true because every prime implicant of a m to 1 multiplexor has a size given by $\lceil \log_2 m \rceil + 1$. In [5] it was observed that the area complexity ($\mathcal{A}_{mux}$) is approximately exponential in the complexity measure. Hence it follows that:

$$\mathcal{A}_{mux} \propto 2^{\lceil \log_2 m \rceil} \qquad (3)$$

Let C_i denote the number of control inputs in a prime implicant P_i. Then define C_{on} to be the average number of control inputs in a prime implicant belonging to the on-set of $\hat{f}$, so that:

$$C_{on} = \frac{\sum_{i=1}^{K_{on}} C_i}{K_{on}} \qquad (4)$$

where K_{on} is the number of prime implicants in the on-set of $\hat{f}$. Similarly, one can define C_{off}. From the above discussion it follows that C_{on} and C_{off} can be used to measure the area contribution of the multiplexor to an optimal area implementation of $\hat{f}$. Notice that the optimal implementation of $\hat{f}$ will contain a (implicit) reduced multiplexor whose area depends on the smaller of C_{on} and C_{off}. Thus, we can model this area contribution, in a fashion analogous to equation (3), as:

$$\mathcal{A}(\hat{f}) - \mathcal{A}(f) \propto 2^{\min\{C_{on}, C_{off}\}} \qquad (5)$$

It then follows from equations (3) and (5) that:

$$\alpha = 2^{\min\{C_{on}, C_{off}\} - \lceil \log_2 m \rceil} \qquad (6)$$

It must be noted that α can be computed with minimal effort from the prime implicants of $\hat{f}$, and once α is available, $\mathcal{A}(f)$ can be computed using (2).

3. High-Level Area Estimation Flow

The transformation, as stated in the previous section, does not place any restriction on the number of outputs that can be dealt with at a time (m). However, we have observed that in practice there is a trade-off between run time of the area estimation procedure and m. As the value of m increases we observed that the time taken to generate the prime implicants usually increases. However, using too small a value of m can affect the accuracy by overestimating the area, as the sharing between all outputs is not captured. After experimenting with different values of m, it was found that a reasonable choice for the value of m was 16.

Typically, a multi-output Boolean function has outputs with varying support set sizes. Outputs whose support set size is very small, for instance 1, 2 or 3, consume very little area. For these outputs very little area optimization can be done. One can make a reliable area prediction for such outputs without having to resort to the above approach. In fact it was found that an area estimate of two gates for outputs whose support set size is two, and an estimate of three gates for outputs with support set size of three, works very well in practice. As far as outputs with support set size of one are concerned, their contribution to an optimal area implementation depends on whether or not they are realized by inversion of a primary input signal. Those which are realized by inversion are assumed to contribute an area of one gate while the rest are assumed not to contribute to the area. The above approach yields benefits in terms of both run time and accuracy, and has been adopted in our area estimation procedure. The flow diagram for the overall area estimation procedure is given in Fig. 2.

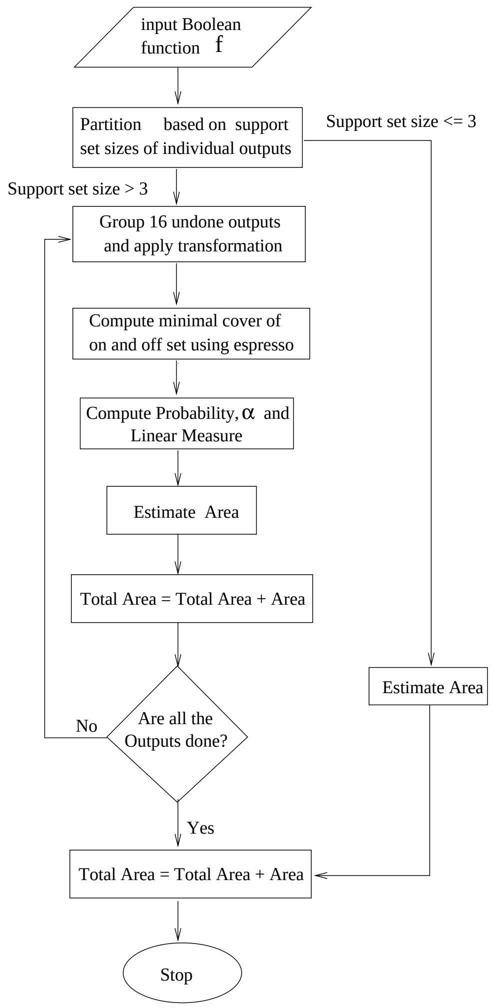

Figure 2. Flow of the Area estimation Procedure.

The area estimation tool reads an input description of f and partitions the function into two sub-functions. One sub-function (f_1) comprises of all outputs whose support set size is less than or equal to three, while the other (f_2), comprises of all outputs whose support set size is greater than three. The partitioning of the network into f_1 and f_2 can be performed by a breadth first search and is fairly inexpensive. We estimate the area of f_1 in the following fashion:

$$\mathcal{A}(f_1) = \beta|f_1^1| + 2|f_1^2| + 3|f_1^3| \qquad (7)$$

Here, $|f_1^1|$ is the number of outputs in f_1 with support set size equal to 1, β is a fraction of these outputs which are realized by inversion of a primary input signal, $|f_1^2|$ is the number of outputs in f_1 with support set size equal to 2, and $|f_1^3|$ is the number of outputs in f_1 with support set size equal to 3. For estimating the area of f_2 we use the transformation based approach described above. Let the outputs of f_2 be grouped into I groups of size sixteen each. Let the Boolean function comprising of the ith group of outputs be g_i. We apply the multiplexor transformation to g_i, and compute α, probability and the *linear measure* of the resultant $\hat{g}_i$. We then compute the area complexity of g_i using (2) and (6). This procedure is repeated until all the outputs have been used up, and the area of f_2 is estimated as:

$$\mathcal{A}(f_2) = \sum_{i=1}^{I} \mathcal{A}(g_i) \qquad (8)$$

Finally, the area of f is computed as:

$$\mathcal{A}(f) = \mathcal{A}(f_1) + \mathcal{A}(f_2) \qquad (9)$$

It must be noted that the proposed area model does not account for area sharing across groups.

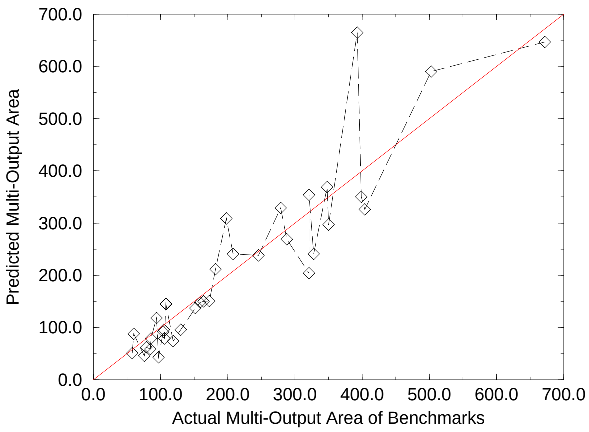

Figure 3. Actual Area versus Predicted Area.

3.1 Empirical Results

The above proposed area model for multi-output functions was tested on several ISCAS-89 and MCNC benchmark circuits. These circuits are listed in Table 1 which, in addition to primary input and output counts, shows the functionality of these benchmarks. These circuits were optimized in SIS using *rugged.script* for optimization, and mapped using the library *lib2.genlib*. The area predicted using the area model was compared with the SIS optimal area.

The performance of the model on all the benchmarks in Table 1, except $s13207^*$ and $s35932$, is shown in Fig. 3. Circuit $s13207^*$ is a modified version of $s13207$, obtained by deleting the primary outputs which contain exclusive-or arrays in them. The SIS-optimal area of $s13207^*$ was 1367. The estimated area for this circuit was 1045. The circuit $s35932$ could not be optimized in SIS in one piece. Hence the circuit was partitioned based on the support set sizes (in a fashion similar to the above discussion) and optimized separately in SIS. The resulting SIS-area that was obtained was 7252. The area estimated by the area estimation tool was 8761.

Table 1.
Characteristics and run-times for
the benchmark function set

CIRCUIT Name	Circuit Function	Inputs	Outputs	CPU Time sec
b9	Logic	41	21	5.7
c8	Logic	28	18	4.9
example2	Logic	85	66	28
frg2	Logic	143	139	268
i7	Logic	199	67	23.1
i8	Logic	133	81	81.5
i6	Logic	138	67	17.5
cht	Logic	47	36	6.5
alu2	ALU	10	6	12.8
alu4	ALU	14	8	104
term1	Logic	34	10	17.4
ttt2	Logic	24	21	6.25
apex6	Logic	135	99	45.3
apex7	Logic	49	37	20.3
x1	Logic	51	35	12.8
x3	Logic	135	99	53
x4	Logic	94	71	28.6
vda	Logic	17	39	39.3
k2	Logic	45	45	170.1
s298	Controller	17	20	4.4
s386	Controller	13	13	4.2
s400	Controller	24	27	8.5
s444	Controller	24	27	8.5
s510	Controller	25	13	6.9
s526	Controller	24	27	10.4
s526n	Controller	24	27	10
s641	Controller	59	43	41.4
s713	Controller	58	42	42.3
s820	Controller	37	24	16.3
s832	Controller	37	24	16.5
s953	Controller	39	52	38.8
s1196	Logic	28	32	163
s1238	Logic	28	32	141
s1494	Controller	27	25	26.8
s1488	Controller	27	25	29.3
s13207	Logic	152	783	212.8
s35932	Logic	1763	1728	942.4

The execution time required by our area estimation tool is also given in Table 1, in cpu seconds on a SUN sparc5 with 24 MB RAM. We compared these run times, on the above benchmarks, with one run of SIS using *script.rugged* followed by SIS technology mapping. The speedup obtained is shown in Fig. 4. The figure shows a speedup between 2x and 24x. Notice that a speedup of 10x was obtained on large benchmarks like $s35932$ and $s13207^*$. It must be kept in mind that the reported SIS time for $s35932$ was obtained after the circuit was partitioned. Strictly speaking the circuit was not completed in SIS. Hence we believe that on large benchmarks the speedups that can be obtained in practice can be significant.

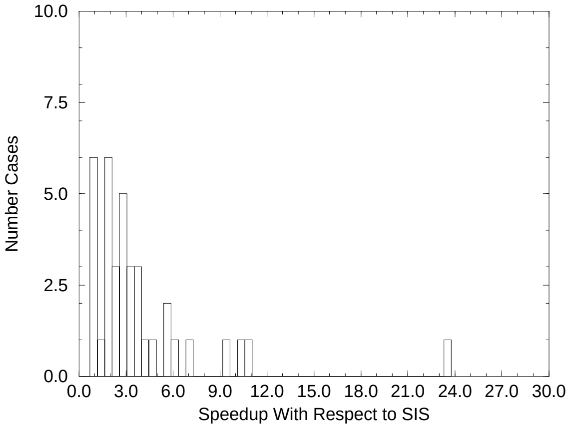

Figure 4. Speed up versus Number of Cases.

4. Estimation of $\mathcal{C}_{avg}$

In order to estimate the power, one needs to estimate not only the area complexity but also $\mathcal{C}_{avg}$, which is the average node capacitance (including interconnect) in a circuit. If $\mathcal{C}_{tot}$ is the total circuit capacitance of an optimal area implementation and $\mathcal{A}$ is the number of gates, then:

$$\mathcal{C}_{avg} = \frac{\mathcal{C}_{tot}}{\mathcal{A}} \tag{10}$$

This quantity depends on the target gate library and on the fan-out structure of the circuit. In order to estimate this, it is assumed that one has access to a few area optimal circuit implementations in the desired target library. This does not appear to be an unreasonable assumption. In this case, an estimate of $\mathcal{C}_{avg}$ can be obtained by performing an average of the $\mathcal{C}_{avg}$ estimates obtained from the area optimal circuit implementations.

In order to test the accuracy of this approach, only a few benchmarks from the benchmark set listed in Table 1 were used to obtain an estimate of $\mathcal{C}_{avg}$. These benchmarks were $s13207^*$, $s35932$, $k2$ and $i8$. This estimated value of $\mathcal{C}_{avg}$ was used to compute $\mathcal{C}_{tot}$, assuming that the exact value of $\mathcal{A}$ was available. The estimated value of $\mathcal{C}_{tot}$ was compared with the true value of $\mathcal{C}_{tot}$, and the results are shown in Fig. 5, which validates the above estimation procedure for $\mathcal{C}_{avg}$.

The estimated value of $\mathcal{C}_{avg}$ was combined with the estimated area complexity of Boolean functions to obtain an estimate of the total capacitance of the Boolean function, $\mathcal{C}_{tot}$. The plot comparing the actual versus predicted values of $\mathcal{C}_{tot}$, when both $\mathcal{A}$ and $\mathcal{C}_{avg}$ are estimated, is shown in Fig. 6.

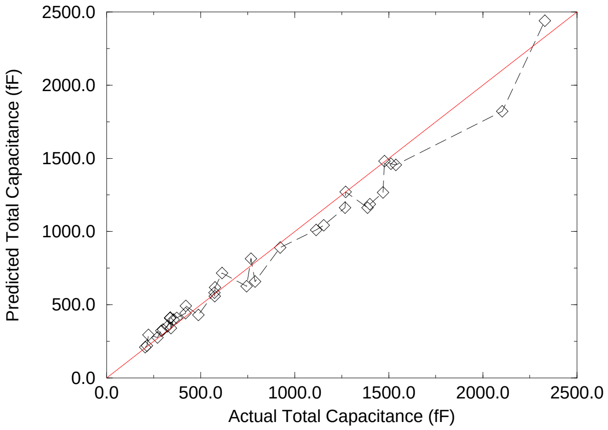

Figure 5. Error between actual and estimated values of $\mathcal{C}_{tot}$ assuming $\mathcal{A}$ is known.

5. High-Level Power Estimation

The area estimate can be used to estimate the power dissipated by a Boolean function, by combining it with average activity estimates [3] and the average node capacitance estimate. We will compare our power estimates to the power dissipated by a gate level optimal area implementation of the Boolean function under two different timing models, namely, the zero-delay model and the general-delay timing model [3]. In the case of the *general-delay* timing model the delays were obtained from a gate library and an event driven simulation was performed.

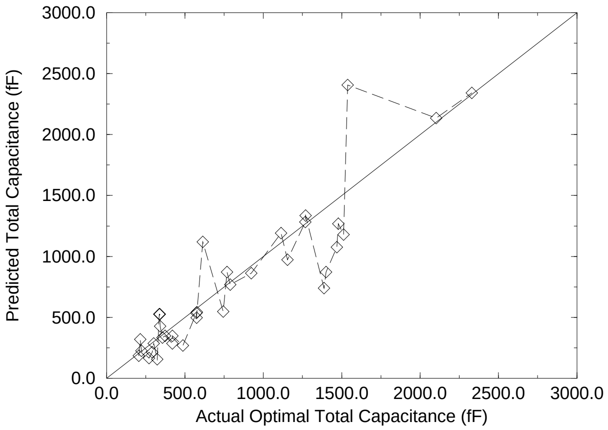

Figure 6. Actual versus Estimated values of $\mathcal{C}_{tot}$ versus Number of Cases.

It must be noted that the activity prediction model [3] does not account for the increase in switching activity due to glitches, as is probably to be expected from a high-level model. Hence it is important to check the accuracy of the high-level power model against the zero-delay simulation results. This is shown in Fig. 7. Since the activity prediction model [3] depends on the input switching statistics of the circuit, we varied the signal probabilities at the circuit inputs from 0.1 to 0.9. Thus, each benchmark circuit is represented by a number of data points in the figure.

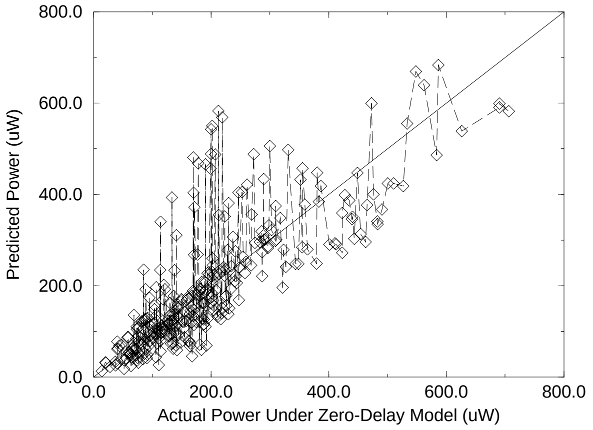

Figure 7. Comparison between actual zero-delay power and predicted power.

For the benchmarks vda and $k2$, the predicted power is significantly different from the actual zero-delay power inspite of the fact that the predicted total capacitance is very close to the true value of total capacitance. This is because of an over-estimation in the average activity of the circuit. The correlation plot between predicted and actual zero-delay power obtained after removal of the power estimates corresponding to these two circuits is shown in Fig. 8. The better agreement in this plot shows that indeed for all but two of the benchmarks considered, the method works rather well. These two circuits are responsible for most of the bad points in Fig. 7.

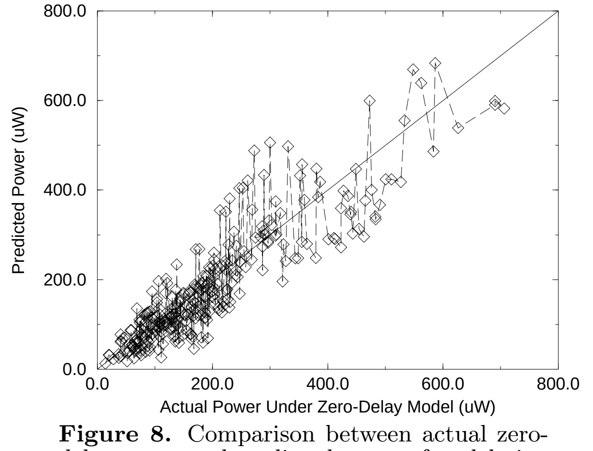

Figure 8. Comparison between actual zero-delay power and predicted power after deletion of points corresponding to vda and $k2$.

We also compared the predicted power against the general-delay simulation results. This is shown in Fig. 9. As is to be expected, the error in the prediction increases. This is due to the possible presence, in the general-delay case, of multiple transitions per cycle at a logic node, i.e., glitches.

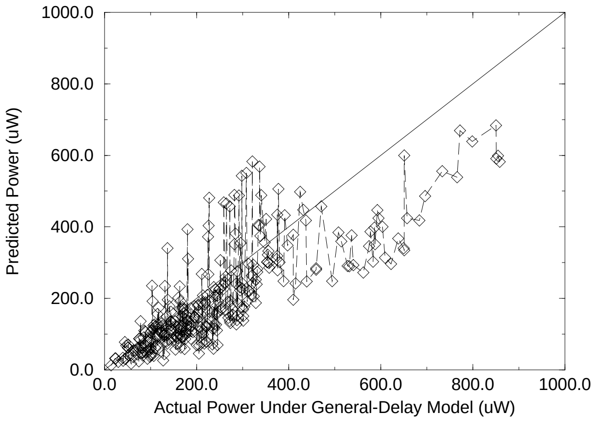

Figure 9. Comparison between actual general-delay power and predicted power.

6. Conclusions

In this paper we presented a new area model to predict the area complexity of multi-output Boolean functions. This was based on transforming the multi-output function at hand to an equivalent single-output function. The advantages of this model is that no additional characterization in necessary beyond that done for single-output functions. Moreover it offers a natural framework to account for sharing occurring in a multi-output function. The predicted capacitance was then combined with average activity estimates [3] to get high level power estimates.

References

[1] F. Najm, "Towards a High-Level Power Estimation Capability," *ACM/IEEE International Symposium on Low-Power Design*, pp. 87–92, 1995.

[2] D. Marculescu, R. Marculescu and M. Pedram, "Information theoretic measures of energy consumption at register transfer level," *International Symposium of Low Power Design*, pp. 81-86, 1995.

[3] M. Nemani and F. Najm, "Towards a high-level power estimation capability," *IEEE Trans. on Computer Aided Design*, vol. 15, no. 6, pp. 588-589, June 1996.

[4] M. Nemani and F. Najm, "High-level power estimation and the area complexity of Boolean functions," *International Symposium of Low Power Electronics and Design*, pp. 329-334, 1996.

[5] M. Nemani and F. Najm, "High-Level Area Prediction for Power Estimation," *Custom Integrated Circuits Conference*, 1997.

[6] A. C-H. Wu, V. Chaiyakul and D. D. Gajski, "Layout area models for high level synthesis," *International Conference on Computer Aided Design*, pp. 34-37, 1991.

[7] F. J. Kurdahi, D. D. Gajski, C. Ramachandran and V. Chaiyakul, "Linking register transfer and physical levels of design," *IEICE Transactions on Information and Systems*, September 1993.

[8] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, New York, NY: McGraw-Hill Inc., 1994.